RFC 683
NIC 32251

R. Clements
BBN

3 April 75 1

Introduction 2

In response to a long-known need for the ability to transfer TENEX paged files over the net via FTP, the TENEX FTP implementation has been extended. 2a

This implementation is an extension to the "OLD" protocol (RFC 354, ff). It was built after useful discussions with Postel, Neigus, et al. I do not mean to imply that they agreed that this implementation is correct, nor for that matter do I feel it is correct. A "correct" implementation will be negotiated and implemented in the "NEW" protocol (RFC 542), if funding ever appears for that task. 2b

The Problem(s) 3

This extension attacks two separate problems: Network reliability and TENEX disk file format's incompatibility with FTP. A checksummed and block-sequence-numbered transmission mode is seriously needed, in my opinion. This mode should also allow data compression. 3a

It is also necessary to handle paged, holey TENEX files. This latter problem, seriously needed for NLS, is the motivation for the current extension. 3b

The former problem requires a new MODE command, if done correctly; probably two MODEs, to allow data compression in addition to checksumming. Actually, I think that is the tip of an iceberg which grows as 2**N for additional sorts of modes, so maybe some mode combination system needs to be dreamed up. Cf the AN, AT, AC, EN, ET, EC TYPEs. Also, one should be able to use MODE B and MODE C together (NEW protocol) to gain both the compression and restart facilities if one wanted. 3c

The second problem, TENEX files, are probably a new kind of STRUcture. However, it should be possible to send a paper tape to a disk file, or vice versa, with the transfer looking like a paged file; therefore perhaps we are dealing with a data representation TYPE. This argument is a bit strained, though, so a paged STRUcture is quite likely correct. I admit to feeling very unsure about what is a MODE, what is a TYPE and what is a STRUcture. 3d

The (Incorrect) choices made 4

Having decided that new MODEs and STRUctures were needed, I instead implemented the whole thing as a single new TYPE. After all, I rationalize, checksumming the data on the network (MODE) and representing the data in the processing system as a checksummed TYPE are really just a matter of where you draw the imaginary line between the net and the data. Also, a single new TYPE command reduced the size of the surgery required on the FTP user and server programs. 4a

Implementation details 5

The name of the new TYPE is "XTP". I propose this as a standard for all the Key Letter class of FTP commands: the "X" stands for "experimental" -- agreed on between cooperating sites. The letter after the "X" is signed out from the protocol deity by an implementor for a given system. In this case, "T" is for TENEX. Subsequent letter(s) distinguish among possibly multiple private values of the FTP command. Here "P" is "Paged" type. 5a

TYPE XTP is only implemented for STRU F, BYTE 36, and MODE S. Information of TYPE XTP is transfered in chunks (I intentionally avoid the words RECORD and BLOCK) which consist of a header and some data. The data in a chunk may be part of the data portion of the file being transfered, or it may be the FDB (File Descriptor Block) associated with the file. 5b

Diversion: the TENEX Disk File 6

For those not familiar with the TENEX file system, a brief dissertation is included here to make the rest of the implementation meaningful. 6a

A TENEX disk file consists of four things: a pathname, a page table, a (possibly empty) set of pages, and a set of attributes. 6b

The pathname is specified in the RETR or STOR verb. It includes the directory name, file name, file name extension, and version number. 6c

The page table contains up to 2**18 entries. Each entry may be EMPTY, or may point to a page. If it is not empty, there are also some page-specific access bits; not all pages of a file need have the same access protection. 6d

A page is a contiguous set of 512 words of 36 bits each.

The attributes of the file, in the FDB, contain such things as creation time, write time, read time, writer's byte-size, end of file pointer, count of reads and writes, backup system tape numbers, etc. 6e

Note that there is NO requirement that pages in the page table be contiguous. There may be empty page table slots between occupied ones. Also, the end of file pointer is simply a number. There is no requirement that it in fact point at the "last" datum in the file. Ordinary sequential I/O calls in TENEX will cause the end of file pointer to be left after the last datum written, but other operations may cause it not to be so, if a particular programming system so requires. 6f

In fact both of these special cases, "holey" files and end-of-file pointers not at the end of the file, occur with NLS data files. These files were the motivation for the new TYPE. 6g

Meanwhile, back at the implementation,... 7

Each chunk of information has a header. The first byte, which is the first word (since TYPE XTP is only implemented for BYTE 36) of the chunk is a small number, currently 6, which is the number of following words which are still in the header. Next come those six words, and then come some data words. 7a

The six header words are:

Word 1: a checksum.

This is a one's complement sum (magnitude and end-around carry) of the six header words and the following data words (but not the leading "6" itself). The sum of all words including the checksum must come out + or - zero.

Word 2: A sequence number.

The first chunk is number 1, the second is number 2, etc.

Word 3: NDW,

the number of data words in this chunk, following the header. Thus the total length of the chunk is 1 (the word containing NHEAD) + NHEAD +NDW. The checksum checks all but the first of these.

Word 4: Page number.

If the data is a disk file page, this is the number of that page in the file's page map. Empty pages (holes) in the file are simply not sent. Note that a hole is NOT the same as a page of zeroes.

Word 5: ACCESS.

The access bits associated with the page in the file's page map. (This full word quantity is put into AC2 of an SPACS by the program reading from net to disk.)

Word 6: TYPE.

A code for what type of chunk this is. Currently, only type zero for a data page, and type -3 for an FDB are sent.

After the header are NDW data words. NDW is currently either 1000 octal for a data page or 25 octal for an FDB. Trailing zeroes in a disk file page will soon be discarded, making NDW less than 1000 in that case. The receiving portions of FTP server and user will accept these shortened pages. The sender doesn't happen to send them that way yet. 7b

Verification is performed such that an error is reported if either: 7c

The checksum fails,

The sequence number is not correct,

NDW is unreasonable for the given chunk type, or

The network file ends at some point other than immediately following the data portion of an FDB chunk.

Closing comments 8

This FTP server and user are in operation on all the BBN systems and at some other sites -- the user being more widely distributed since fewer sites have made local modifications to the user process. 8a

I believe the issues of checksumming and sequencing should be addressed for the "NEW" protocol. I hope the dissertation on TENEX files has been useful to users of other systems. It may explain my lack of comprehension of the "record" concept, for example. A TENEX file is just a bunch of words pointed to by a page table. If those words contain CRLF's, fine -- but that doesn't mean "record" to TENEX. I think this RFC also points out clearly that net data transfers are implemented like the layers of an onion: some characters are packaged into a line. Some lines are packaged into a file. The file is broken into other managable units for transmission. Those units have compression applied to them. The units may be flagged by restart markers (has anyone actually done that?). The compressed units may be checksummed, sequence numbered, date-and-time stamped, and flagged special delivery. On the other end, the process is reversed. Perhaps MODE, TYPE, and STRU don't really adequately describe the situation. This RFC was written to allow implementors to interface with the new FTP server at TENEX sites which install it. It is also really a request for comments on some of these other issues. 8b

RFC 683, NIC 32251

FTPSRV - TENEX Extensions for Paged Files

R. Clements - BBN - 3 April 1975

RECEIVED
APR 14 1975

Postel

RECEIVED
APR 14 1975

1 Introduction

In response to a long-known need for the ability to transfer TENEX paged files over the net via FTP, the TENEX FTP implementation has been extended.

This implementation is an extension to the "OLD" protocol (RFC 354, ff). It was built after useful discussions with Postel, Neigus, et al. I do not mean to imply that they agreed that this implementation is correct, nor for that matter do I feel it is correct. A "correct" implementation will be negotiated and implemented in the "NEW" protocol (RFC 542), if funding ever appears for that task.

2 The Problem(s)

This extension attacks two separate problems: Network reliability and TENEX disk file format's incompatibility with FTP. A checksummed and block-sequence-numbered transmission mode is seriously needed, in my opinion. This mode should also allow data compression.

It is also necessary to handle paged, holey TENEX files. This latter problem, seriously needed for NLS, is the motivation for the current extension.

The former problem requires a new MODE command, if done correctly; probably two MODEs, to allow data compression in addition to checksumming. Actually, I think that is the tip of an iceberg which grows as 2**N for additional sorts of modes, so maybe some mode combination system needs to be dreamed up. Cf the AN, AT, AC, EN, ET, EC TYPEs. Also, one should be able to use MODE B and MODE C together (NEW protocol) to gain both the compression and restart facilities if one wanted.

The second problem, TENEX files, are probably a new kind of STRUcture. However, it should be possible to send a paper tape to a disk file, or vice versa, with the transfer looking like a paged file; therefore perhaps we are dealing with a data representation TYPE. This argument is a bit strained, though, so a paged STRUcture is quite likely correct. I admit to feeling very unsure about what is a MODE,

what is a TYPE and what is a STRUcture.

3 The (Incorrect) choices made

Having decided that new MODEs and STRUctures were needed, I instead implemented the whole thing as a single new TYPE. After all, I rationalize, checksumming the data on the network (MODE) and representing the data in the processing system as a checksummed TYPE are really just a matter of where you draw the imaginary line between the net and the data. Also, a single new TYPE command reduced the size of the surgery required on the FTP user and server programs.

4 Implementation details

The name of the new TYPE is "XTP": I propose this as a standard for all the Key Letter class of FTP commands: the "X" stands for "experimental" -- agreed on between cooperating sites. The letter after the "X" is signed out from the protocol deity by an implementor for a given system. In this case, "T" is for TENEX. Subsequent letter(s) distinguish among possibly multiple private values of the FTP command. Here "P" is "Paged" type.

TYPE XTP is only implemented for STRU F, BYTE 36, and MODE S.

Information of TYPE XTP is transfered in chunks (I intentionally avoid the words RECORD and BLOCK) which consist of a header and some data. The data in a chunk may be part of the data portion of the file being transfered, or it may be the FDB (File Descriptor Block) associated with the file.

5 Diversion: the TENEX Disk File

For those not familiar with the TENEX file system, a brief dissertation is included here to make the rest of the implementation meaningful.

A TENEX disk file consists of four things: a pathname, a page table, a (possibly empty) set of pages, and a set of attributes.

The pathname is specified in the RETR or STOR verb. It includes the directory name, file name, file name extension, and version number.

The page table contains up to 2**18 entries. Each entry may be EMPTY, or may point to a page. If it is not empty, there are also some page-specific access bits; not all pages of a file need have the same access protection.

A page is a contiguous set of 512 words of 36 bits each.

The attributes of the file, in the FDB, contain such things as creation time, write time, read time, writer's byte-size, end of file pointer, count of reads and writes, backup system tape numbers, etc.

Note that there is NO requirement that pages in the page table be contiguous. There may be empty page table slots between occupied ones. Also, the end of file pointer is simply a number. There is no requirement that it in fact point at the "last" datum in the file. Ordinary sequential I/O calls in TENEX will cause the end of file pointer to be left after the last datum written, but other operations may cause it not to be so, if a particular programming system so requires.

In fact both of these special cases, "holey" files and end-of-file pointers not at the end of the file, occur with NLS data files. These files were the motivation for the new TYPE.

6 Meanwhile, back at the implementation,...

Each chunk of information has a header. The first byte, which is the first word (since TYPE XTP is only implemented for BYTE 36) of the chunk is a small number, currently 6, which is the number of following words which are still in the header. Next come those six words, and then come some data words.

The six header words are:

Word 1: a checksum. This is a one's complement sum (magnitude and end-around carry) of the six header words and the following data words (but not the leading "6" itself). The sum of all words including the checksum must come out + or - zero.

Word 2: A sequence number. The first chunk is number 1, the second is number 2, etc.

Word 3: NDW, the number of data words in this chunk, following the header. Thus the total length of the chunk is 1 (the word containing NHEAD) + NHEAD +NDW. The checksum checks all but the first of these.

Word 4: Page number. If the data is a disk file page, this is the number of that page in the file's page map. Empty pages (holes) in the file are simply not sent. Note that a hole is NOT the same as a page of zeroes.

Word 5: ACCESS. The access bits associated with the page in the file's page map. (This full word quantity is put into AC2 of an SPACS by the program reading from net to disk.)

Word 6: TYPE. A code for what type of chunk this is. Currently, only type zero for a data page, and type -3 for an FDB are sent.

After the header are NDW data words. NDW is currently either 1000 octal for a data page or 25 octal for an FDB. Trailing zeroes in a disk file page will soon be discarded, making NDW less than 1000 in that case. The receiving portions of FTP server and user will accept these shortened pages. The sender doesn't happen to send them that

way yet.

Verification is performed such that an error is reported if either:

The checksum fails,
The sequence number is not correct,
NDW is unreasonable for the given chunk type, or
The network file ends at some point other than immediately following the data portion of an FDB chunk.

7 Closing comments

This FTP server and user are in operation on all the BBN systems and at some other sites -- the user being more widely distributed since fewer sites have made local modifications to the user process.

I believe the issues of checksumming and sequencing should be addressed for the "NEW" protocol. I hope the dissertation on TENEX files has been useful to users of other systems. It may explain my lack of comprehension of the "record" concept, for example. A TENEX file is just a bunch of words pointed to by a page table. If those words contain CRLF's, fine -- but that doesn't mean "record" to TENEX. I think this RFC also points out clearly that net data transfers are implemented like the layers of an onion: some characters are packaged into a line. Some lines are packaged into a file. The file is broken into other managable units for transmission. Those units have compression applied to them. The units may be flagged by restart markers (has anyone actually done that?). The compressed units may be checksummed, sequence numbered, date-and-time stamped, and flagged special delivery. On the other end, the process is reversed. Perhaps MODE, TYPE, and STRU don't really adequately describe the situation. This RFC was written to allow implementors to interface with the new FTP server at TENEX sites which install it. It is also really a request for comments on some of these other issues.